(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 750 428 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
27.12.1996 Bulletin 1996/52

(51) Int Cl.[6]: H04N 7/32

(21) Application number: 96304626.3

(22) Date of filing: 21.06.1996

(84) Designated Contracting States:
DE FR GB

(30) Priority: 22.06.1995 JP 156146/95
22.06.1995 JP 156147/95
14.07.1995 JP 178708/95

(71) Applicant: CANON KABUSHIKI KAISHA
Tokyo (JP)

(72) Inventors:
- Kajiwara, Hiroshi
Tokyo (JP)
- Yoshida, Takashi
Tokyo (JP)
- Kirabayashi, Yasuji
Tokyo (JP)

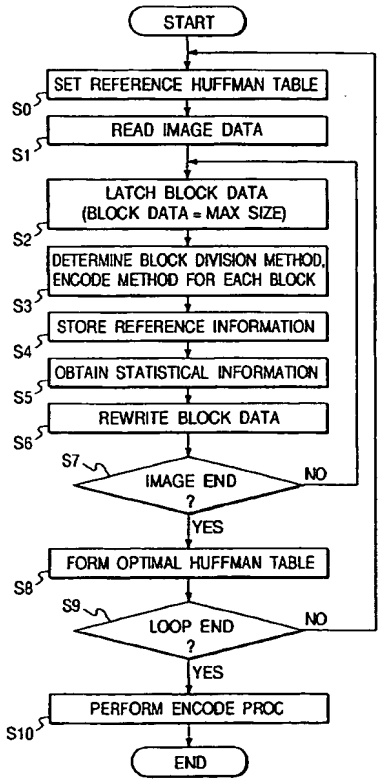
(74) Representative:
Beresford, Keith Denis Lewis et al
BERESFORD & Co.
2-5 Warwick Court
High Holborn
London WC1R 5DJ (GB)

(54) Image processing apparatus and method

(57) An encoding error by the encoding and the decoding of an image is controlled within a predetermined range by an image processing apparatus which includes an input unit for inputting image data, a prediction unit for predicting pixel values indicated by the image data, a generation unit for generating pixel values derived by converting the pixel values of the image data within a range of ±e based on the prediction result by said prediction unit and an encoding unit for encoding parameters relating to the pixel values generated by said generation unit based on the prediction result by said prediction unit.

An image having a gradient in the values of block pixels is encoded at a high efficiency by an image processing apparatus which includes a division unit for dividing an image into block pixels to be encoded, a generation unit for generating parameters defining a plane based on values of first block pixels and values indicating positions of the block pixels to be encoded and a plane encoding unit for encoding difference values between the values of the block pixels indicated by the plane defined by the parameters and the values of the encoded block pixels.

FIG. 1



EP 0 750 428 A2

Description

The present invention relates to image processing apparatus and method for encoding an image.

An image encoding method is classified into two major classes, an information preservation type encoding method in which an original image is completely restored when decoded and an information-non-reservation type encoding in which an error from the original image is included.

As the information preservation type encoding method, a DPCM encoding method has been known. In a common DPCM encoding, a predicted value of a pixel to be encoded is calculated based on encoded pixels around the pixel to be encoded, that is, a left pixel (A), an upper pixel (B) and a lift diagonal pixel (C) by using a formula (A+B-C), for example, and a difference between the result and the value of the pixel to be encoded is encoded.

In the information non-reservation type encoding method, it is not possible to completely restore the original image but it allows a high compression ratio. Therefore, when a large amount of image data is handled or an error is not critical, the information non-reservation type encoding system is frequently used. A JPEG baseline system is one of the information non-reservation type encoding methods.

In prior art image compression, a block code has been frequently used. In a conversion encoding which includes many irrevocable encoding, a block is a unit of encoding.

The concept of predicting a pixel value by a plane has been adopted in a commonly used DPCM. For example, for a pixel under consideration, a predicted value is calculated by (a+b-c) where a is a left adjacent pixel value, b is an upper pixel value and c is a left upper pixel value.

In general, since most portions of the image are formed by simple brilliance gradients, the effect of using the above prediction is great.

In the prior art prediction encoding system, the value of pixel to be encoded is predicted from values of surrounding pixels and a difference between the predicted value and the value of the pixel under consideration is entropy-coded and transmitted. In this system, the amount of codes generated is influenced by the prediction system and an optimum prediction system differed from area to area. Accordingly, in order to efficiently apply this system, it is common to use a method in which the image is divided into blocks and an optimal prediction system is selected for each block and a code is generated by adding information for specifying the prediction system.

The method for selecting the prediction system for each block is effective when the optimal prediction method differs from block to block but the transmission of the added information is not effective when the prediction system is uniform in a large area such as an image background area.

In the prior art image encoding, since the amount of encoded data is simply controlled, it is not possible to encode an image in which a certain degree of error between an actual pixel value of the image to be encoded and a pixel value of the encoded and decoded version of the original image is permitted (an allowable error range), at a high efficiency and within a predetermined error of the pixel value.

A digital image which is actually handled is obtained by reading a printed document or a silver halide photograph by an image scanner or a CCD sensor, and in many cases, random noises are included over an entire area of the image. From experience, in an 8-bit tone brilliance data, random noise having a maximum amplitude of 3 is included. When the prediction is made based on the prior art method of a+b-c, a noise which has a larger dispersion than a dispersion value of the noise is included in the predicted value and hence the efficiency of the encoding is not enhanced.

It is an aim of one aspect of the present invention to encode image data at a high compression ratio while controlling an error of a pixel value.

In accordance with a preferred embodiment of the present invention, there is provided an image processing apparatus comprising input means for inputting image data, prediction means for predicting pixel values indicated by the image data; generation means for generating pixel values derived by converting the pixel values of the image data within a range of ±e based on the prediction result by the prediction means and encoding means for encoding parameters relating to the pixel values generated by the generation means based on the prediction result by the prediction means. It is an aim of another aspect of the present invention to encode multivalue data at a high compression ratio while controlling an error of a pixel value.

It is an aim of a further aspect of the present invention to provide an encoding method which eliminates the problems in the prior art and allows encoding of data block by block, particularly a high efficiency reversible encoding of gradient image data. accordance with a preferred embodiment of the present invention, there is provided an image processing apparatus comprising division means for dividing an image into block pixels to be encoded, generation means for generating parameters defining a plane based on values of first block pixels and values indicating positions of the block pixels to be encoded and plane encoding means for encoding difference values between the values of the block pixels indicated by the plane defined by the parameters and the values of the encoded block pixels.

Embodiments of the present invention will now be described with reference to the accompanying drawings, in which:

Fig. 1 shows a flow of a process in a first form of implementation of one embodiment of the present

invention;
Figs. 2A and 2B show types of an encoding method and a positional relation of pixels;
Fig. 3 shows a portion of image data in the encoded data;
Fig. 4 illustrates a Huffman table to encode c0';
Fig. 5 shows a method for scanning run length in a #8 encoding method;
Fig. 6 show a manner of dividing a block to be processed into sub-blocks;
Fig. 7 shows a flow of an S3 process;
Fig. 8 which is composed of Figs. 8A to 8C shows a flow of S21, S25 and S32 processes;
Fig. 9 shows a maximum block division method and encoded data format;
Fig. 10 shows a manner of converting a difference value to a corrected difference value;
Fig. 11 shows a flow of an S10 encoding process;
Fig. 12 shows a circuit configuration to implement one embodiment of the present invention;
Fig. 13 shows a circuit configuration to implement the first form of implementation;
Fig. 14 shows a block diagram of an encoding apparatus of a first embodiment of the present invention;
Fig. 15 shows position of surrounding pixels a, b and c for a pixel under consideration;
Fig. 16 illustrates block generation; and
Fig. 17 shows a block diagram of a plane encoding circuit.

[First Embodiment]

Fig. 1 shows a flow of a process of a first embodiment of the present invention. In Fig. 1, S0 denotes a step to set a reference Huffman table to be described later, S1 denotes a step to read image data, S2 denotes a step to read block data from the image data at a maximum block size (16×16 pixels in the present embodiment), S3 denotes a step to determine a block division method and an encoding method, S4 denotes a step to store the block division method and the encoding method determined in the step S3 as reference information, S5 denotes a step to calculate statistics of frequency of occurrence of difference value (prediction error) to be described later which is outputted based on the reference information, S6 denotes a step to update the block data read in the step S2 to virtual block data to be described later, S7 denotes a step to determine an end point of an image, S8 denotes a step to generate an optimal Huffman table based on the difference value statistics calculated in the step S5, S9 denotes a step to determine a loop end and S10 denotes a step to encode the entire image based on the reference information.

The present embodiment is now briefly explained. The encoding in the present embodiment is not a non-reversible encoding but an encoding method in which an error between a pixel value of an actual image and a pixel value of an encoded image is admitted within a range of ±e at maximum. This is referred to as a near lossless encoding.

It is assumed here that a maximum size of block to be encoded is 16×16 pixels and the encoding is performed at one of 16×16 pixels, 8×8 pixels and 4×4 pixels. For the encoding method for each block, six types of prediction encoding methods (#0 to #5), two types of plane encoding methods (#6 and #7) and a run length encoding method (#8) which may be used only when the block is represented in binary form may be selected (see Figs. 2A and 2B).

The nine types of encoding methods used for the encoding the block are briefly explained. Fig. 3 shows formats of code data by the respective encoding methods.

The #0 to #5 are for the prediction encoding method in which the predicted value is calculated using surrounding pixels A, B and C shown in Fig. 2A by the formulas of A+B-C, B, A, B+(A-C)/2, A+(B-C)/2, (A+B+C)/3, and a difference between the pixel under consideration and the predicted value is Huffman-encoded. The Huffman table is prepared for each block size and it is encoded by using a separate Huffman table for each block size.

The #6 and #7 are for the plane encoding method. In the #6, a plane which approximates a pixel value of the block is determined by $P(x, y)=ax+by+c0$, where c0 is a left upper corner pixel value in the block, and the values of a, b and c0 and a difference between the approximate plane and the pixel value in the block are Huffman-encoded.

If a and b are below a predetermined level, a difference c0' from the previously appeared c0 is Huffman-encoded for c0, and if they are above the predetermined level, the c0 value itself is encoded. A flag is inserted after a, b and c0 so that the difference data is not added when all of the reference values in the block are zero.

The #7 is used when all pixel values in the block are equal, that is, when it is a plane of a=b=c, and the c0' described above is Huffman-encoded. Fig. 4 shows a Huffman table which is used when the c0' is encoded in the #6 and #7. In Fig. 4, S denotes a sign bit which is 0 or 1 depending on the sign of the difference value c0. When the c0 is no smaller than 14 or no larger than -14, the difference value is not used and the value of c0 is added after the ESC code.

The #8 is for the run length encoding method and it is used only when there are only two pixel values in the block. For example, assuming that a higher brilliance pixel represent white and a lower brilliance pixel represents black, the pixels in the block are scanned in the order shown in Fig. 5, converted to a run length code and it is encoded. For the encoding, predetermined white (high brilliance) run and black (low brilliance) run Huffman table are used and the last run in the block is substituted by the EOB code.

Referring to Fig. 1, a flow of process in the present

embodiment is now explained in detail.

In the step S0, the reference Huffman table to be used in encoding the block is set. A Huffman table prepared based on statistical information of predicted errors generated by prediction encoding determined for a plurality of images or by other method is read and a 16×16 pixel prediction coding Huffman table, a 8×8 pixel prediction encoding Huffman table, a 4×4 pixel prediction encoding Huffman table and a plane encoding Huffman table are set.

Four counters to be used when those Huffman tables are newly prepared in the step S8, that is, a 16×16 prediction encoding counter, a 8×8 pixel prediction encoding counter, a 4×4 pixel prediction encoding counter and a plane encoding counter are all initialized to zero.

A Huffman table for Huffman-encoding the information indicating the encoding method (#1 to #8) is prepared for each block size, an encoding counter to be used in newly preparing the Huffman tables is provided and it is also initialized to zero.

In the step S1, the image data to be encoded is read onto an encoding memory.

In the step S2, the block data (maximum block size 16×16 pixels) are sequentially read from the image to be encoded. The upper and left lines of the read block are also read as reference areas for the prediction encoding.

In the step S3, for the maximum size block (16×16 pixels) read in the step S1, the block division method to further dividing the block to sub-block (blocks to be encoded) appropriate for the encoding and the encoding method (#0 to #8) appropriate for the each block to be encoded are determined and the pixel values in the maximum block are updated within an error of ±e from the actual pixel values. In this manner, the encoding can be performed with a small difference value (prediction error). By updating the pixel value within the range of ±e so that the difference values have only several values, the amount of information of the difference values can be reduced. The process of the step S3 is described below in detail.

The routine of the step S3 has a reversible structure and a total amount of codes when the maximum block or a cubic block derived by dividing the maximum block several times (which is refereed to as the block to be processed and the size thereof differs depending on the flow of process) is sub-divided (into four blocks) is compared with an amount of codes when such division is not made to determine whether the block is to be divided or not (see Fig. 6), and the reference information including the block division information and the encoding method as well as the amount of codes when the block selected by the reference information is encoded are outputted.

Fig. 7 shows a detailed flow of the iterative process used when the encoding method of the maximum block (the block to be processed) and the division method are selected in the step S3.

In Fig. 7, S30 denotes a step to determine the size of the block to be processed. If the block to be processed is the minimum block size (4×4 pixels), the process proceeds to S33, other wise the process proceeds to S20 and S23.

S20 denotes a step to copy the block to be processed to generate a block A. The block to be processed is copied to the block A in the memory area for the iterative process. S21 denotes a step to select the optimal encoding method to minimize the amount of codes LA when the block A is encoded. Detail of the selection method will be described later.

S22 denotes a step to output the amount of codes LA generated when the encoding method selected in S21 is used.

S23 denotes a step to copy the block to be processed to generate a block B. S24 denotes a step to divide the block B into four sub-blocks and S25 denotes a step to select an optimal encoding method for each sub-block as is done in the step S21.

S26 denotes a step to output the total amount of codes which is a sum of amounts of codes when the four sub-blocks are encoded by using the encoding methods selected in the step S25.

In S27, the LA outputted in the step S22 is compare with the LB outputted in the step S29 and if LA<LB, the process proceeds to S28, and if LA≥LB, the process proceeds to S29.

S28 denotes a step to determine the size of the block to be processed in the block A as the size of the block to be processed, and the iterative process to determine the block division method for the range of the block to be processed is terminated. In S28 of the iterative process, the block A, that is, the block to be encoded is updated to the pixel values (within the error range of ±e) of a virtual block to be described later generated in selecting the encoding method in the step S21. The process proceeds to S31 to generate the block division information and the encoding method selected in the step S21 and the amount of codes LA as the reference information.

S29 denotes a step to copy the block B comprising four sub-blocks to the range of the block to be processed used at the beginning of the iterative process. The four sub-blocks are substituted by the blocks to be processed and the iterative process to determine the block division method is iterated from the beginning.

In S32, an optimal encoding method of the block to be processed having the minimum block size is selected. In S33, the amount of codes L is outputted. In S34, the block to be processed is updated by the pixel values of the virtual block to be generated later which is generated in the step S32. In S35, the optimal encoding method and the amount of codes L for the block are generated.

So far, the flow of the iterative process used when the encoding method of the maximum block (or the block to be processed) and the block division method are se-

lected in the step S3 has been described.

The selection of the optimal encoding method in the steps S21, S25 and S32 is now explained in detail.

Figs. 8A to 8C show a flow to select the optimal encoding method. The amounts of codes of all selectable encoding methods are estimated to conduct the selection of the optimal encoding method.

Prior to the estimation of the amounts of codes, the block data is updated to a form appropriate for each method. In Fig. 8A, S40 denotes a step to initialize the amount of codes Ln(0) generated by the encoding method #0, S41 denotes a step to determine a predicted value A+B-C, S42 denotes a step to update the pixel values in the block within the error range of ±e, S43 denotes a step to add one pixel of Huffman code length for the difference value to Ln(0), and S44 denotes a step to determine an end point of the block. When the amount of codes in encoding the block is generated, the process proceeds to S61.

The above precess is referred to mode 0 and mode 1 to mode 5 correspond to the processes for the encoding methods #1 to #5 which are identical to the above process.

S45 denotes a step to initialize the encoding methods #6 and #7. A flag (FLAG) indicating whether the approximate plane and the pixel values in the block completely match or not is prepared and it is initially set to 1 (complete match). The parameter Ln(6) indicating a code length for #6 is set to 0 and a parameter Ln(7) indicating a code length for #7 is set to a value which is not minimum of the Ln(0) to Ln(8). S46 denotes a step to determine whether the entire block is rendered to the same pixel value or not by changing the pixel values in the block within the error range ±e, S47 denotes a step to determine the block approximate plane P(x, y) = ax + by + c0 of the #6 described above, S48 denotes a step for each pixel to update the pixel value within the error range ±e to generate a pixel value for forming the virtual block and S49 denotes a step to add the Huffman code length for the difference value between the generated pixel value and the approximate plane P(x, y) to the Ln (6). S50 denotes a step to determine whether the difference value (prediction error) is 0 or not. When there is even one pixel value which is different from that of the approximate plane, the FLAG is set to 0 in S51. In S52, the end point of the block is determined.

S53 denotes a step to determine the value of the FLAG. If FLAG = 0, the amount of codes of the difference value for each pixel generated in S49 and the amount of codes parameters a, b and c0 (c0') indicating the approximate plane P(x, y) are added to produce Ln(6) (S54), and the process proceeds to S61.

If FLAG = 1 (the approximate plane and the pixel values of the block completely match), only the information of the approximate plane need be encoded and the amount of codes of the parameters a, b, c0 (c0') are substituted by Ln(6) (S55), and the process proceeds to S61.

When the value of Ln(6) is determined, Ln(7) is set to a value which is not minimum of Ln(0) to Ln(8).

In S56, the amount of codes of c0 (c0') is set to the value of Ln(7) and Ln(6) is set to a value which is not minimum of Ln(0) to Ln(8). In S64, the block data is updated (to generate the virtual block) within the range of ±e such that all pixel values in the block are rendered to c0 and the process proceeds to S61.

S57 denotes a step to determine whether the block is represented by two pixel values or not. If it is represented by the two pixel values, the process proceeds to S58, otherwise Ln(8) is set to a value which is not minimum of Ln(0) to Ln(8), and the process proceeds to S61.

S58 denotes a step to convert the pixel values to a run length. The amount of codes Ln(8) generated when the run length is encoded is determined and the process proceeds to S61.

S61 denotes a step to compare the values of Ln(0) to Ln(8) to select a minimum value and the corresponding optimal encoding method. S62 denotes a step to store the virtual block corresponding to the encoding method selected in S61 of the virtual blocks generated by updating the pixel values within the range of ±e in the respective modes (mode 0 to mode 8), and S63 denotes a step to output the optimal encoding method and the amount of codes.

The process before the step S61 determines the amounts of codes for the respective methods. This process is conducted after the block to be processed (or the sub-block) is copied in the separate memory area. Accordingly, the updating of the block data when the optimal encoding method is selected does not affect to the original block to be processed.

The steps S40 to S44 are now explained in detail. The steps S40 to S44 determine the amount of codes Ln(0) generated by the #0.

S41 to S44 denote a loop to raster-process the pixels in the block sequentially from the left upper and the processed pixels (the pixels updated in S42) are substituted by the original pixel value as the reference pixels of the pixels under consideration. When the reference pixels, (A, B, C) are out of the area of the 16×16 pixel block read in S2, the pixel values of the block data updated in S6 before the loop precess (S2 to S7) of the previous maximum block is used.

In the step S40, Ln(0) is set to 0. In the step S41, the predicted value P=A+B-C is generated based on the pixel value X under consideration and the surrounding pixel values A, B and C and the difference value D=X-P is calculated. In the step S42, a corrected difference value D' is determined by a formula (1) and the pixel under consideration is updated to D'+P. In the formula (1), ⌊x⌋ represents omission of digits after the decimal point.

$$D'=\lfloor(D+\alpha)/(2e+1)\rfloor\times(2e+1) \quad (1)$$

$$\alpha=(D/(|D|)\times e \quad (2)$$

Fig. 11 illustrates a corrected difference value D' when e=3 and the difference value D=4. As shown in Fig. 11, since the number of the difference values D'(-7, 0, 7) is smaller than the number of difference values D (-10 to 10), the information indicating the difference values is significantly reduced. Further, the possibility of the complete matching of the difference values and the predicted values is high.

In the present embodiment, the pixel value to be corrected (the corrected pixel value) X'=X+3. In the step S43, the difference between the updated corrected pixel value X' and the predicted value P, that is, the corrected difference value D' is Huffman-coded by the Huffman table for the size of the block to be processed (or the sub-block) and the generated code length is added to Ln(0). In the step S44, whether the process has been completed to the end point of the block or not is determined, and if it is completed, the process proceeds to a step S61, and if it is not completed, the next pixel is considered and the process starting from the step S41 is repeated. In this manner, the amount of codes Ln(0) generated when the encoding method #0 is used is determined. For the encoding methods #1 to #5, only the prediction methods are different and the methods for determining the amount of generated codes Ln(I to 5) are same as that for the #0 and the detailed description thereof is omitted.

The steps S45 to S56 and S64 are now explained in detail. S45 denotes a step to initialize the variables for the #6 and #7. The amount of codes Ln(6) generated by the #6 is set to 0 and the amount of codes Ln(7) generated by the #7 is set to a sufficiently large value. A variable FLAG which is used to determine whether all difference values are 0 or not in the #6 is set to 1. In a step S46, a maximum value (max) and a minimum value (min) of the pixel values in the block are determined, and if max-min is smaller than 2e+1, the process branches to a process for the encoding method for the #7 (steps S56 et seq), otherwise the process proceeds to a process for the encoding method #6 (steps S47 et seq).

In a step S47, an approximate plane is determined from the pixel values in the block. The steps S48 to S52 sequentially process the pixel values in the block. In a step S48, the difference value D=X-P is determined from the pixel values X under consideration in the block and the value P corresponding to the position of the pixel under consideration on the approximate plane determined in the step S47, the corrected difference value D' is calculated in the same manner as that in the step S42 and the pixel under consideration is updated to D'+P. In a step S49, the difference value between the updated corrected pixel value X' and the predicted value P is Huffman-encoded by the Huffman table for the plane encoding and the code length thereof is added to Ln(6).

In a step S50, whether the difference value between the corrected pixel value X' and the predicted value P is 0 or not, and if it is not 0, the FLAG is set to 0 in a step S51. In a step S52, the end point of the block is determined, and if it is not the end point, the process starting from the step S48 is continued for the next pixel in the block. S53 denotes a step to determine the value of the FLAG. If it is 0, the process proceeds to S54, otherwise the process proceeds to S55. In the step S54, the amount of codes of a, b and c0 plus one bit for the FLAG is added to the Ln(6). In the step S55, the amount of codes of a, b and c0 plus one bit for the FLAG is substituted by a new Ln(6).

If it is determined that the encoding method #7 is available in the step S46, a quotient of (max-min)/2 is set to c0 in a step 556 and the amount of codes when c0 is encoded is substituted for the Ln(7) and a sufficiently large value is set to the Ln(6).

In a step S64, all pixel values in the block are updated. By this process, the amounts of codes Ln(6) and Ln(7) when the encoding methods #6 and #7 are used are determined.

The steps S57 to S60 are now explained in detail. S57 denotes a step to determine whether the encoding method #8 is available or not. The pixel values in the block are examined and if two pixel values appear, the process proceeds to S58, otherwise the process proceeds to S50.

In S50, a sufficiently large value is set to the Ln(8). In a step S58, the block is scanned in the order shown in Fig. 5 while a higher brilliance value of the two pixel values of the block is set to white and a lower brilliance value is set to black to convert it to white-and-black run length data. In a step S59, the amount of codes to represent the two appearing pixel values and the amount of codes when the run length data is Huffman-encoded by using the Huffman table exclusively prepared for the #8 are added to determine the amount of generated codes Ln(8) for the #8.

In this manner, the amounts of generated codes Ln (0) to Ln(8) for the encoding methods #0 to #8 are determined, and in a step S61, a minimum amount of codes which is the minimum of the above is determined and a sum of the minimum amount of codes and amount of codes indicating the encoding method is stored as the amount of code I of the block (one of the four amounts of codes comprising the amount of code LA or the total amount of code LB in Fig. 7, or the amount of code L) and the information representing the encoding method number (i) is also stored.

In a step S62, the virtual block generated in using the encoding method #i is stored. The virtual block is used to determine the block to be encoded (update the original image values) in the steps S28 and S34, and it is finally used to update the block data in the step S6.

In a step S63, the number i and the amount of codes I are outputted and the optimal encoding method selection process (corresponding to S21, S25 and S32 of Fig.

7) is terminated. The description of the selection of the optimal encoding method in the steps S21, S25 and S32 is thus completed.

In this manner, the block division information and the optimal encoding method in the step S3 have been determined.

Fig. 9 shows an example of the encoding method for each of the divided blocks to be encoded. The reference information based on Fig. 5 is shown in Fig. 10. This completes the detailed description of the step S3.

In a step S4, the reference information generated in S3 is stored in the memory.

In a step S5, for the encoding method number i selected for each of the blocks to be encoded, the numbers of times of occurrence of the encoding methods #0 to #8 are counted by encoding number counters provided one for each block size, and if it is one of #0 to #6, the number of times of occurrence of the corrected difference value D' when encoded by that method is added to the counter corresponding to the encoding method (the prediction encoding counter for #0 to #5, and the plane encoding counter for #6).

In a step S6, the pixel values of the block read in the step S2 are updated by using the data of the block to be encoded which were updated in the virtual block in the step S3 (based on S28 and S34).

In a step S7, the end point of the image is determined, and if it is not the end point of the image, the process staring from the step S2 is conducted to the next block, and if it is the end point, the process proceeds to a step S8.

The loop precess of S2 to S7 is the process to update the block data (virtual block) for each block to generate the image data. When the reference pixels (A, B, C or c0) used in the selection process of the encoding method in the step S3 are out of the area of the 16×16 pixel block read in the step S2, the pixel values of the block data updated in the step S6 before the previous loop precess are used.

In a step S8, based on the values of the 16×16 pixel prediction encoding counter, the 8×8 pixel prediction encoding counter, the 4×4 pixel prediction encoding counter and the plane prediction counter, the 16×16 pixel prediction encoding Huffman table, the 8×8 pixels prediction encoding Huffman table, the 4×4 pixel prediction encoding Huffman table and the plane encoding Huffman table are reconstructed. Further, the numbers of times of occurrence of the optimal encoding methods #0 to #8 are counted by the encoding method number counters provided for the respective block sizes, and the Huffman table for encoding the encoding method numbers are reconstructed in accordance with the statistics of the counts.

In a step S9, the end of the optimization processing loop (S0 to S8) is determined, and when a predetermined number of loops is completed, the process proceeds to the step S9, otherwise the process proceeds to the step S0. In this case, the S0 in the initial optimization processing loop and the step S0 in the second and subsequent optimization steps are different. In the S0 of the second and subsequent loops, the optimal Huffman table prepared in the previous step S8 is substituted for the reference Huffman table.

In a step S10, the image data updated in the step S6 is actually encoded by referring the stored reference information. Fig. 12 shows a flow of the encoding process in the step S10. In Fig. 12, S80 denotes a step to encode the Huffman table, S81 denotes a step to read block data of the maximum size (16×16 pixels) from the image, S82 denotes a step to read the reference data corresponding to the block data, S83 denotes a step to encode the block data in accordance with the reference information and S84 denotes a step to determine the end point of the block. Detailed description follows.

In the step S80, for the 16×16 pixel encoding number Huffman table, the 8×8 pixel encoding number Huffman table, the 4×4 encoding number Huffman table, the 16×16 pixel prediction encoding Huffman table, the 8×8 prediction encoding Huffman table, the 4×4 prediction encoding Huffman table and the plane encoding Huffman table, the information indicating the structures of the respective Huffman tables is outputted as the Huffman table information.

In the step S81, the block data are sequentially read with the maximum block size from the image data. In the step S82, the reference information for the block data is read from the memory.

In the step S83, the reference information and the data derived by encoding the block data of the maximum block size in accordance with the reference information are outputted as the encoded data sequence.

In the step S84, the end point of the image is determined, and if an unprocessed block is present, the process starting from the step S81 is conducted for the next block.

In this manner, the encoded data sequence for the image data is generated and outputted.

This completes the description of the first embodiment.

[Modification of First Embodiment]

The present invention is not limited to the above embodiment. For example, while the block is divided with the maximum block size of 16×16 pixels in the above embodiment, the block size may be set to 128×128 or larger pixel.

Further, while the block is divided into four same size sub-blocks in the above embodiment, the block size may be variable.

In the first embodiment, the block is divided based on the amount of codes of the block to be processed and the amounts of codes of the sub-blocks derived by dividing the block to determine the block to be encoded. Alternatively, the amount of codes of all blocks from the maximum block (16×16 pixels) to the minimum block

(4×4 block) may be determined and then the block may be divided based on the amount of all codes.

Further, the amount of codes of the minimum block may be first determined, this block is regarded as the sub-block, a temporary block division method and the amount of codes are determined based on the total amount of codes of the four sub-blocks (4×4 pixels) as shown in Fig. 6 and the amount of codes of the block (8×8 pixels) of the combined area, a new temporary block division method is determined based on the total amount of codes of the four determined areas (8×8 pixels) and the amount of codes of a larger block (16×16 pixels) comprising the determined area (8×8 pixels), and it is repeated until the maximum block size is reached to determine the block to be encoded.

Further, while the prediction encoding method, the plane encoding method and the run length encoding method are used as the selectable block encoding method in the above embodiment, the application method may be changed by changing the prediction method or other encoding method such as the Marcof model encoding method may be used. While the Huffman encoding is uses as the entropy encoding method in the above embodiment, other entropy encoding method such as arithmetic encoding method may be used.

In S6, the block data need not be updated but the value of the prediction error (generated in S3) when the image by the updated block data is encoded may be stored. In this case, in S10, instead of reencoding by using the updated block data, the encoding may be made by using the prediction error (corrected difference value) determined in the step S3. However, when the difference values for the respective pixels are stored, the circuit configuration and the process to generate the reference pixel values are complex compared to the process in which the prediction encoding is conducted by referring the pixel values (A, B and C) in S41 of Fig. 8A. Accordingly, in the first embodiment, the method of storing the corrected pixel value is used rather than the method for storing the corrected difference value.

Fig. 13 shows an example of the apparatus for conducting the encoding process of the first embodiment. Numeral 1 denotes a CPU control unit which can conduct the encoding and decoding of the image data by the software processing, numeral 2 denotes an image input unit for reading the image data, numeral 3 denotes a printer unit for printing the image data, numeral 5 denotes a communication control unit for transmitting the data encoded by the CPU control unit 1 or the encoded data transmitted from an external equipment and numeral 6 denotes an image memory for storing the image information.

[Second Embodiment]

Fig. 14 shows a block diagram of a second embodiment of the present invention. In Fig. 14 numeral 100 denotes an image input unit for generating image data such as an image reader, a TV camera or a host computer, numeral 101 denotes a block generation circuit for blocking image data for each pixel, numeral 103 denotes a plane encoding circuit, numeral 104 denotes a buffer for adjusting timing, numeral 105 denotes a sub-block generation circuit for dividing the block in the block generation circuit into sub-blocks, numerals 106, 107, 108 and 109 denote prediction encoding circuit for conducting the prediction encoding in different methods, numeral 110 denotes a selector, numeral 111 denotes an information addition circuit, numeral 112 denotes a buffer for adjusting the timing, numeral 113 denotes a selector, numeral 114 denotes an information addition circuit, numerals 115, 116, 117 and 118 denote signal lines and numeral 119 denotes an output unit for outputting the encoded data to an external memory and an external equipment.

Fig. 15 shows positions of surrounding pixels a, b and c used by the prediction encoding circuits 106, 107, 108 and 109. In Fig. 15, x represents a pixel under consideration and a, b and c are defined by the relative positions to x.

First, the image data to be encoded is stored in the memory 101 from the image input unit 100. The block generation circuit 102 sequentially reads the image data by 16×16 pixel block and stores them. As shown in Fig. 3, the data of the left and upper pixels of the 16×16 pixel block are also read as shown in Fig. 3 and they are stored. The plane encoding circuit 103 calculates a plane $\alpha x+\beta y+\gamma$ (x, y=0,...,15) which proximates the 16×16 pixel block stored in the block generation circuit 102 and Huffman-encodes the values of the parameters α, β and γ and the difference value between the proximate plane and the pixels values in the block and stores them in the buffer 104. Detail of the plane encoding circuit 103 will be described later.

On the other hand, the sub-block generation circuit 105 sequentially reads the 8×8 pixel blocks from the 16×16 pixel block stored in the block generation circuit 102 and stores them. The data of the left and upper pixels of the 8×8 pixel block are also read and stored. The prediction encoding circuit 106 calculates x-a for each pixel in the 8×8 pixel sub-block stored in the sub-block generation circuit 105 and Huffman-encodes it. Similarly, the prediction encoding circuit 107 Huffman-encodes x-b, the prediction encoding circuit 108 Huffman-encodes x-c and the prediction encoding circuit 109 Huffman-encodes x-a-b+c. The selector 110 selects the encoded data of the smallest amount of codes of the encoded data outputted from the prediction encoding circuits 106, 107, 108 and 109, outputs the code sequence thereof to the signal line 116 and outputs 2-bit identification data I to the signal line 115 to identify the selected prediction encoding circuit. The information addition circuit 111 adds the identification data I from the signal line 115 to the head of the code sequence of the signal line 116 and stores them in the buffer 112. This process is conducted for the four pixel sub-blocks so that one block

of codes of the 16×16 pixels are stored in the buffer 112.

The selector 113 compares the amount of codes of the buffer 104 and the buffer 112, selects the smaller one and outputs it to the signal line 118. It also outputs one-bit identification data II for identifying the selected buffer 104 or 112 to the signal line 117. The information addition circuit 114 adds the identification data II from the signal line 117 to the head of the code sequence and outputs them to the output unit 119.

The above process is repeated until the last block stored in the memory 101 is reached to generate the code.

Fig. 17 shows a configuration of the plane encoding circuit 103.

In Fig. 17, numeral 201 denotes a plane calculation circuit, numeral 202 denotes a plane data generation circuit, numeral 203 denotes a buffer, numeral 204 denotes a differentiation circuit, numeral 205 denotes a parameter encoding circuit, numeral 206 denotes a Huffman encoding circuit and numeral 207 denotes a multiplexor.

In the present embodiment, the image data to be encoded is an 8-bit/pixel tonality image.

The encoding is conducted block by block. The 16×16 pixel block data from the block generation circuit 201 is sent to the plane calculation circuit 201 and the buffer 203 through the signal lines 208 and 209.

The plane calculation circuit 201 calculates the parameters a, b, cθ and m defining the plane in a manner to be described later and sends them to the plane generation circuit 202 and the parameter encoding circuit 205 through 210, 211 and 212. The plane generation circuit 202 generates the plane data in accordance with the received parameters and outputs it to 214. A difference between it and the input block image data from 213 is calculated for each pixel in the block by the differentiation circuit 204 and the plane prediction error value is sent to the Huffman encoding circuit 206 for each pixel. The Huffman encoding circuit 206 allocates the Huffman code in accordance with the built-in Huffman table and outputs the code data to 217.

On the other hand, the parameter inputted to the parameter encoding circuit 205 from the signal line 211 is encoded by an appropriate number of bits and it is sent to the multiplexor 207 from 216. The multiplexor 207 rearrange the parameter code sent from 216 and the difference code sent from 217 in a predetermined sequence to generate a series of bit stream which is outputted to the buffer 104.

An operation of the plane calculation circuit 201 is now explained. It is assumed that the pixel value of the left upper pixel of the input N×N (N=16 in the present embodiment) pixels is cθ. When the pixel value in the pixel coordinate (x, y) in the block is dxy, the following statistics is taken.

$$A = \sum_{j}^{N-1}\sum_{i}^{N-1} i^2 = \sum_{j}^{N-1}\sum_{i}^{N-1} j^2$$

$$B = \sum_{j}^{N-1}\sum_{i}^{N-1} 2\ i\ j$$

$$C = \sum_{j}^{N-1}\sum_{i}^{N-1} 2\ (C\theta - d_{ij})\ i$$

$$D = \sum_{j}^{N-1}\sum_{i}^{N-1} 2\ (C\theta - d_{ij})\ j$$

By using those formulas,

$$a=(BD-2AC)/(4A^2-B^2)$$

$$b=(BC-2AD)/(4A^2-B^2)$$

are calculated and an integer m=min(⌈ max(|a|, |b|) ⌉, 16) is determined, where | | indicates an absolute value and ⌈ ⌉ indicates round-up, max( ) indicates a larger one and min( ) indicates a smaller one. Thus, m assumes 1 to 16. Further, a and b are normalized and quantized by using m.

$$a= \lfloor (a/m+1)128 \rfloor$$

$$b=\lfloor (b/m+1)128 \rfloor$$

where ⌊ ⌋ indicates omission.

Thus, a and b are quantized to values of 0 to 255.

When the initially calculated m is 16, a and/or b can assume a value smaller than 0 or larger than 255 and hence it is clipped to keep the range of 0 to 255. After a and b are rendered to 8-bit values, 1 is subtracted from m so that m is rendered to a 4-bit value of 0 to 15.

The result from the above operation is a parameter which defines a best fit plane based on a minimum square error when the plane passing through the left upper pixel of the block is given by Z=ax+by+Cθ and the position of the left top pixel is (0, 0, Cθ).

When all parameters have been calculated, the plane calculator 201 outputs 8-bit values for a, b and Cθ

and a 4-bit value for m.

An operation of the plane generation circuit 202 is now explained.

The plane generation unit 202 receives the plane parameters a, b, Cθ and m explained above from 106 to generate the prediction block. Namely, a' and b' are calculated by

$$a'=(m+1)(a-128)256$$

$$b'=(m+1)(b-128)256$$

and the pixel value d_{ij} of the coordinate (i, j) (where i and j are integers between 0 and N-1) is generated by

$$d_{ij}=L\ (C\theta+a'\ i+b'\ j+0.5)J$$

The block data generated in this manner is outputted to 214.

While all pixels in the block are encoded in the above embodiment, (1) the block may be divided into sub-blocks and the code indicating whether it is the sub-block to be encoded or not may be added for each sub-block, or (2) a plane corresponding to one-bit block may be provided to determine whether it is the sub-block to be encoded or not. The method briefly described in this supplement is particularly effective to a relatively complex portion in the image.

In accordance with the plane encoding described above, the prediction method which does least cause the reduction of the encoding efficiency is attained for the simple brilliance gradient portion bearing noise which occupies a large portion of the actual image, and the encoding efficiency to the entire image is significantly improved over the prior art.

[Modification of Second Embodiment]

The present invention is not limited to the above embodiment. For example, while four values a, b, c and a+b-c are recited as the prediction values used in the prediction encoding in the above embodiment, any other combination which is constructed from the referable pixels may be used. While the Huffman encoding is used together with the plane encoding and the prediction encoding in the above embodiment, other entropy encoding system such as the arithmetic encoding may be used. The plane encoding is not limited to the embodiment in which α, β and γ of the proximate plane αx+βy+γ are encoded as the plane defining information but any values which can specify the plane may be used. Alternatively, instead of the plane, a relatively simple curved surface may be used.

The image encoding apparatus of the second embodiment comprises block division means for dividing the image data into blocks, plane encoding means for determining the proximate plane for each divided block and encoding the information specifying the proximate plane and the difference value between the proximate plane and the block, means for sub-dividing the block into sub-blocks, prediction encoding means for selecting the optimal prediction system for each sub-block and encoding the information specifying the selected prediction system and the error by the selected prediction system, means for selecting one of the plane encoding means and the prediction encoding means for each block and means for adding the information for specifying the selected encoding system. Accordingly, the plane encoding for the block having the planar nature and the prediction encoding for each sub-block for other block may be selectively used. Thus, the amount of generated codes can be suppressed.

By making the block size of the plane encoding to be larger than the block size of the prediction encoding, the efficient encoding is attained by utilizing the characteristic of the plane encoding.

By extracting the information specifying the proximate plane as the coefficients of the operation formula, the efficiency of the encoding by block by block is enhanced.

By converting the pixel values in the image or the block, the image data can be encoded with a relatively high efficiency while controlling the error of the pixel values by using the relatively simple circuit configuration and process.

Further, the multi-value data can be encoded at the high efficiency while controlling the error of the pixel values.

Further, since the image data to be encoded is updated prior to the encoding within the range of ±e, the encoding unit can confirm the decoded image. In this case, if it is after S6, the updated image data may be read and monitored on the display to confirm the decoded image.

The image is divided into blocks, the formula of the proximate plane which proximate the block is calculated for each block and the error from the proximate plane is encoded. Thus, when the original image free from the actual noise is sufficiently proximated by the plane, the dispersion of the noise bearing on the prediction value can be suppressed to the same degree as that of the noise originally included in the original image and the reduction of the encoding efficiency is prevented.

The encoding by using the proximate plane for each block and the encoding by the prediction method selected from the plurality of prediction methods are provided so that the high efficiency encoding is attained even when the multi-value data is reversibly encoded.

The present invention may be modified without departing from the scope of the claims.

Claims

1. An image processing apparatus comprising:

   input means for inputting image data;
   prediction means for predicting pixel values indicated by the image data;
   generation means for generating pixel values derived by converting the pixel values of the image data within a range of ±e based on the prediction result by said prediction means; and
   encoding means for encoding parameters relating to the pixel values generated by said generation means based on the prediction result by said prediction means.

2. An image processing apparatus according to Claim 1, wherein the prediction by said prediction means conducts a plurality of prediction methods.

3. An image processing apparatus according to Claim 1, wherein the parameters relating to the pixel values comprise difference values between the pixel values generated by said generation means and the prediction values predicted by said prediction means.

4. An image processing apparatus according to Claim 1, wherein the pixel values indicated by the image data are represented by multi-value.

5. An image processing apparatus according to Claim 1, wherein the encoding by said encoding means is conducted block by block.

6. An image processing method comprising the steps of:

   inputting image data;
   predicting pixel values indicated by the image data;
   generating pixel values derived by converting the pixel values of the image data within a range of ±e based on the prediction result of said prediction means; and
   encoding parameters relating to the pixel values generated in said generation step based on the prediction result of said prediction means.

7. An image processing apparatus comprising:

   division means for dividing an image into a plurality of block pixels;
   generation means for generating parameters indicating prediction values of the block pixels;
   conversion means for converting the values of the block pixels while controlling differences between the prediction values and the values of the block pixels within a predetermined range; and
   encoding means for encoding the block pixels converted by said conversion means based on the parameters.

8. An image processing apparatus according to Claim 7, wherein the parameters represent a three-dimensional plane, two dimensions of the three dimensions represent a position of the block pixel and one dimension represents a pixel value.

9. An image processing method comprising the steps of:

   dividing an image into a plurality of block pixels;
   generating parameters indicating prediction values of the block pixels;
   converting the values of the block pixels while controlling differences between the prediction values and the values of the block pixels within a predetermined range; and
   encoding the block pixels converted by said conversion step based on the parameters.

10. An image processing apparatus comprising:

    input means for inputting image data;
    prediction means for predicting pixel values indicated by the image data;
    encoding means for encoding difference values between the pixel values indicated by the image data and prediction values predicted by said prediction means; and
    control means for controlling the difference values.

11. An image processing apparatus according to Claim 10, wherein the prediction by said prediction means uses a plurality of prediction methods.

12. An image processing apparatus according to Claim 10, wherein the difference values are set to discrete integers.

13. An image processing apparatus according to Claim 10, wherein the pixel values indicated by the image data are represented by multi-value.

14. An image processing method comprising the steps of:

    inputting image data;
    predicting pixel values indicated by the image data;
    encoding difference values between the pixel values indicated by the image data and prediction values predicted by said prediction step;

and
controlling the difference values.

15. An image processing apparatus comprising:

division means for dividing an image into block pixels to be encoded;
generation means for generating parameters defining a plane based on values of first block pixels and values indicating positions of the block pixels to be encoded; and
plane encoding means for encoding difference values between the value of the block pixels indicated by the plane defined by the parameters and the values of the encoded block pixels.

16. An image processing apparatus according to Claim 15, wherein said plane is a three-dimensional plane.

17. An image processing apparatus according to Claim 15, wherein said plane is a proximate plane based on the values of the encoded pixels.

18. An image processing apparatus according to Claim 15, wherein the parameters are coefficients of a function representing the plane.

19. An image processing apparatus according to Claim 17, wherein said plane is determined based on statistics derived by calculating the values of the encoded block pixels.

20. An image processing apparatus according to Claim 15, further comprising prediction encoding means different from said plane encoding means, wherein said plane encoding means and said prediction means are selectively used.

21. An image processing method comprising the steps of:

dividing an image into block pixels to be encoded;
generating parameters defining a plane based on values of first block pixels and values indicating positions of the block pixels to be encoded; and
encoding difference values between the values of the block pixels indicated by the plane defined by the parameters and the values of the encoded block pixels.

FIG. 1

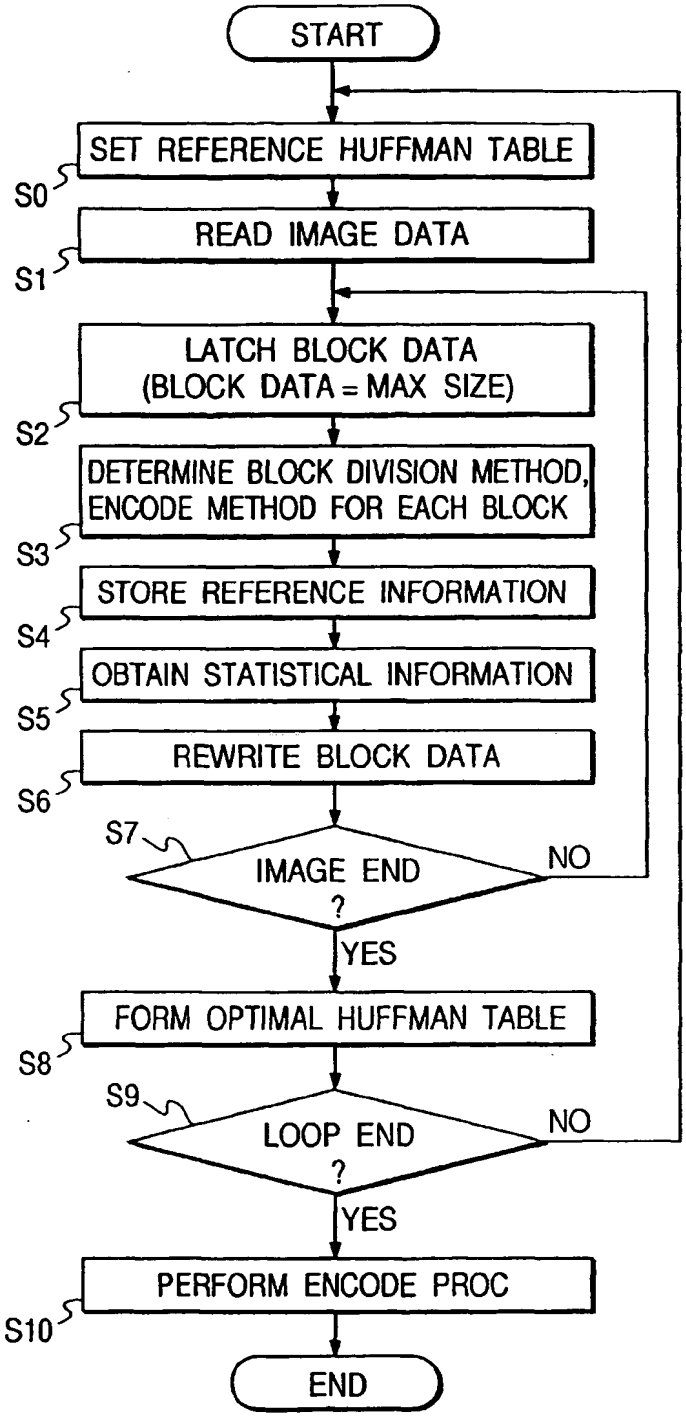

START
SET REFERENCE HUFFMAN TABLE
S0
READ IMAGE DATA
S1
LATCH BLOCK DATA
(BLOCK DATA = MAX SIZE)
S2
DETERMINE BLOCK DIVISION METHOD,
ENCODE METHOD FOR EACH BLOCK
S3
STORE REFERENCE INFORMATION
S4
OBTAIN STATISTICAL INFORMATION
S5
REWRITE BLOCK DATA
S6
S7
IMAGE END
?
NO
YES
FORM OPTIMAL HUFFMAN TABLE
S8
S9
LOOP END
?
NO
YES
PERFORM ENCODE PROC
S10
END

FIG. 2A

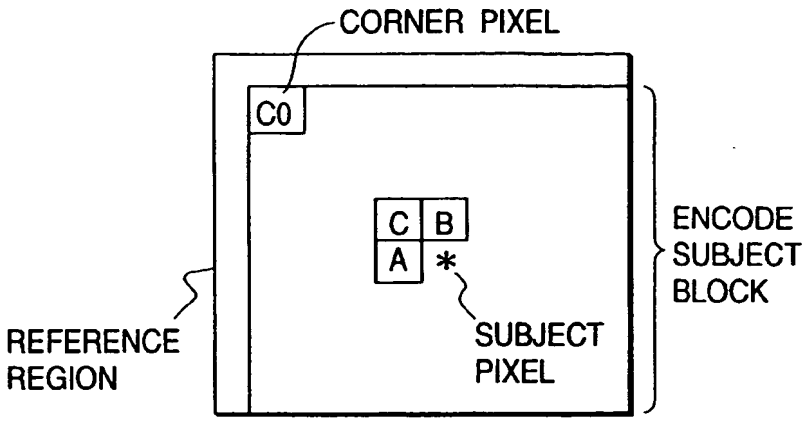


FIG. 2B

NO.	ENCODE METHOD
#0	A+B-C
#1	B
#2	A
#3	B+(A-C)/2
#4	A+(B-C)/2
#5	(A+B+C)/3
#6	PLANE APPROXIMATION MODE
#7	UNIFORM MODE
#8	BINARY MODE

HUFFMAN TABLE BEING COMMON IN EACH BLOCK SIZE

FIG. 3

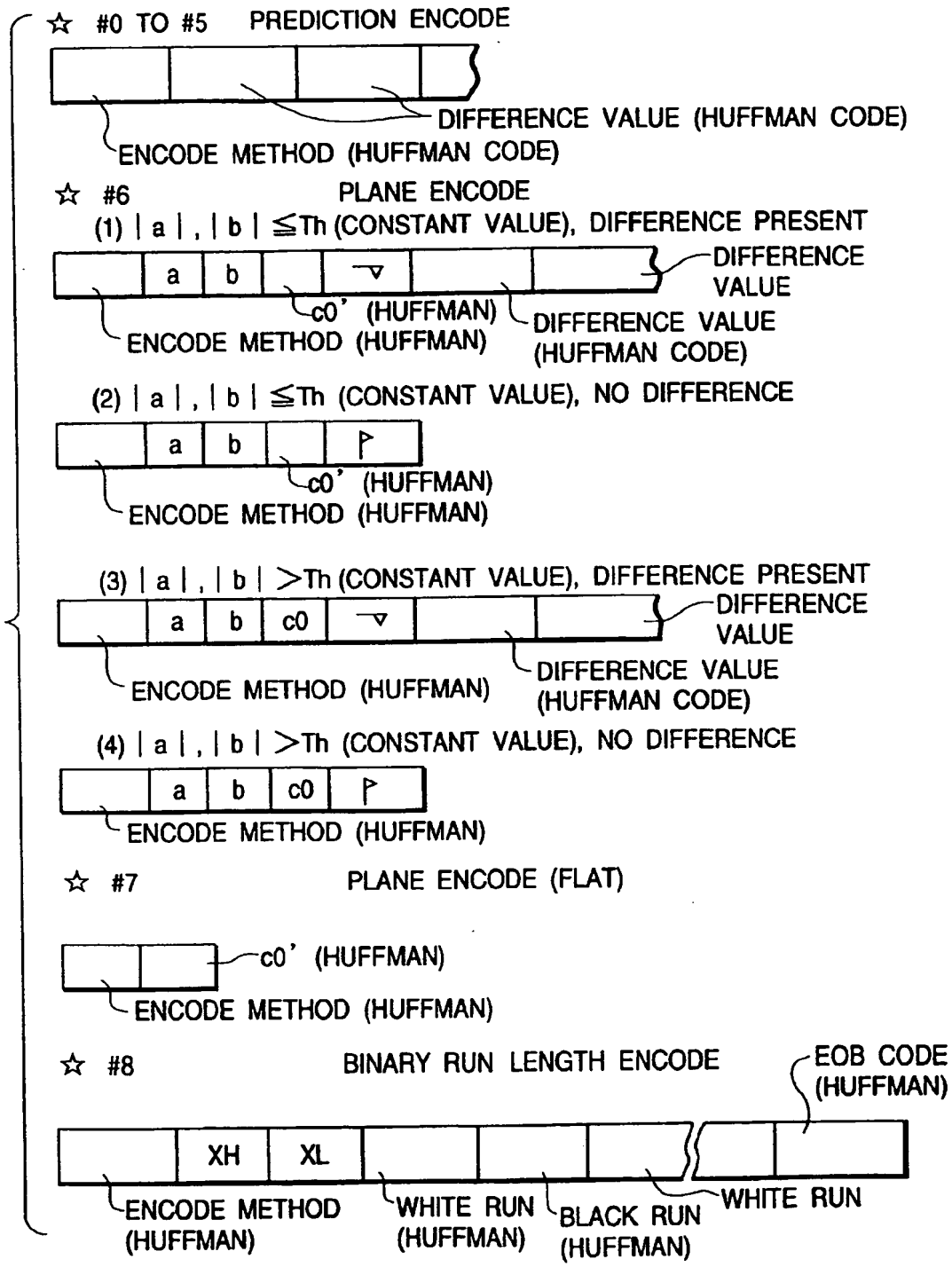
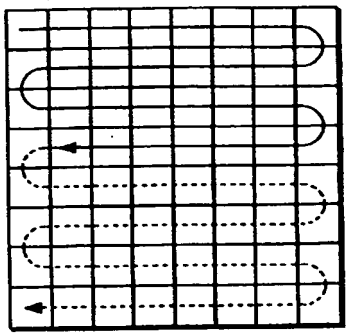

☆ #0 TO #5 PREDICTION ENCODE
DIFFERENCE VALUE (HUFFMAN CODE)
ENCODE METHOD (HUFFMAN CODE)
☆ #6 PLANE ENCODE
(1) | a | , | b | ≦Th (CONSTANT VALUE), DIFFERENCE PRESENT
a
b
DIFFERENCE VALUE
c0' (HUFFMAN)
ENCODE METHOD (HUFFMAN)
DIFFERENCE VALUE (HUFFMAN CODE)
(2) | a | , | b | ≦Th (CONSTANT VALUE), NO DIFFERENCE
a
b
c0' (HUFFMAN)
ENCODE METHOD (HUFFMAN)
(3) | a | , | b | >Th (CONSTANT VALUE), DIFFERENCE PRESENT
a
b
c0
DIFFERENCE VALUE
ENCODE METHOD (HUFFMAN)
DIFFERENCE VALUE (HUFFMAN CODE)
(4) | a | , | b | >Th (CONSTANT VALUE), NO DIFFERENCE
a
b
c0
ENCODE METHOD (HUFFMAN)
☆ #7 PLANE ENCODE (FLAT)
c0' (HUFFMAN)
ENCODE METHOD (HUFFMAN)
☆ #8 BINARY RUN LENGTH ENCODE
EOB CODE (HUFFMAN)
XH
XL
ENCODE METHOD (HUFFMAN)
WHITE RUN (HUFFMAN)
BLACK RUN (HUFFMAN)
WHITE RUN

FIG. 4

c0 DIFFERENCE (c0')	HUFFMAN CODE
0	0
-1, 1	100 S
-2, 2	101 S
-3, 3	110 S
-4, 4	111000 S
-5, 5	111001 S
-6, 6	111010 S
-7, 7	111011 S
-8, 8	111100 S
-9, 9	111101 S
-10, 10	111110 S
-11, 11	11111100 S
-12, 12	11111101 S
-13, 13	11111110 S
ESC	11111111

HUFFMAN CODE TABLE OF c0'

FIG. 5



SQUARE BLOCK

FIG. 6

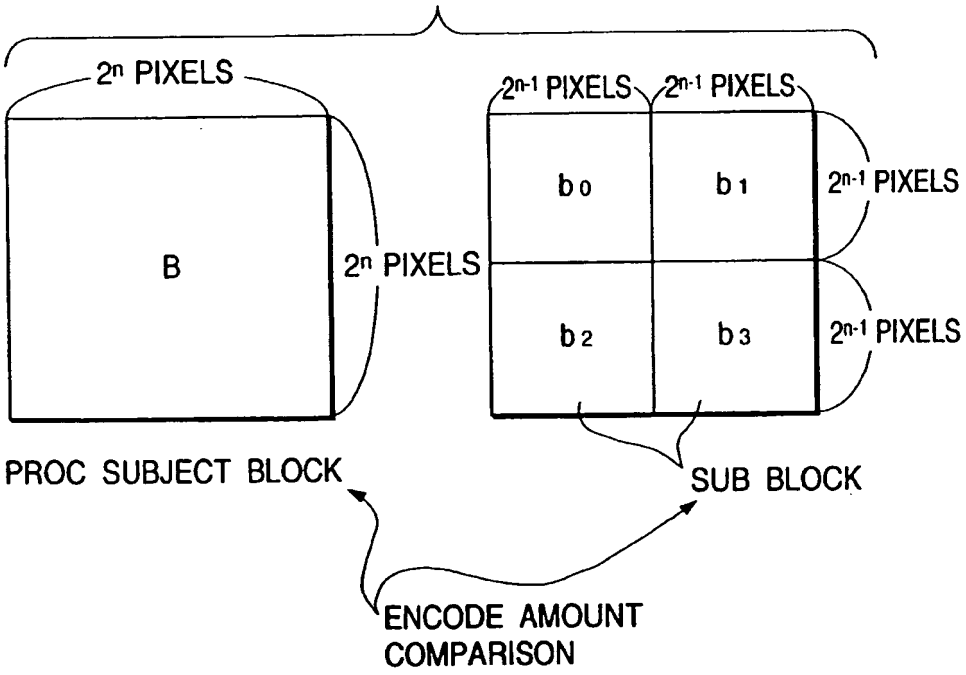


FIG. 7

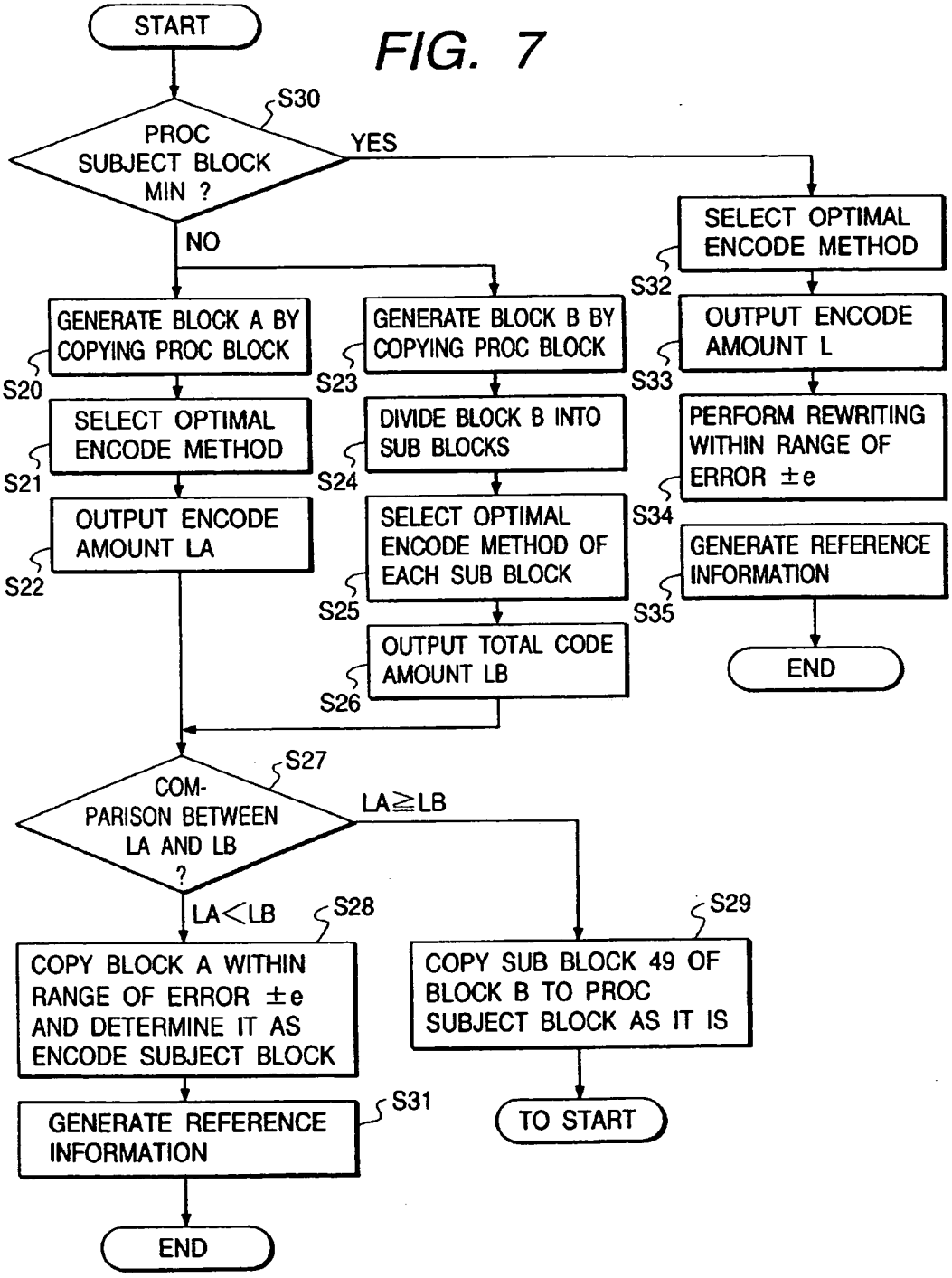

START
S30
PROC SUBJECT BLOCK MIN ?
YES
NO
SELECT OPTIMAL ENCODE METHOD
S32
OUTPUT ENCODE AMOUNT L
S33
PERFORM REWRITING WITHIN RANGE OF ERROR ±e
S34
GENERATE REFERENCE INFORMATION
S35
END
GENERATE BLOCK A BY COPYING PROC BLOCK
S20
SELECT OPTIMAL ENCODE METHOD
S21
OUTPUT ENCODE AMOUNT LA
S22
GENERATE BLOCK B BY COPYING PROC BLOCK
S23
DIVIDE BLOCK B INTO SUB BLOCKS
S24
SELECT OPTIMAL ENCODE METHOD OF EACH SUB BLOCK
S25
OUTPUT TOTAL CODE AMOUNT LB
S26
S27
COM-PARISON BETWEEN LA AND LB ?
LA≧LB
LA<LB
S28
COPY BLOCK A WITHIN RANGE OF ERROR ±e AND DETERMINE IT AS ENCODE SUBJECT BLOCK
S29
COPY SUB BLOCK 49 OF BLOCK B TO PROC SUBJECT BLOCK AS IT IS
TO START
S31
GENERATE REFERENCE INFORMATION
END

FIG. 8

FIG. 8A	FIG. 8B	FIG. 8C

FIG. 8A

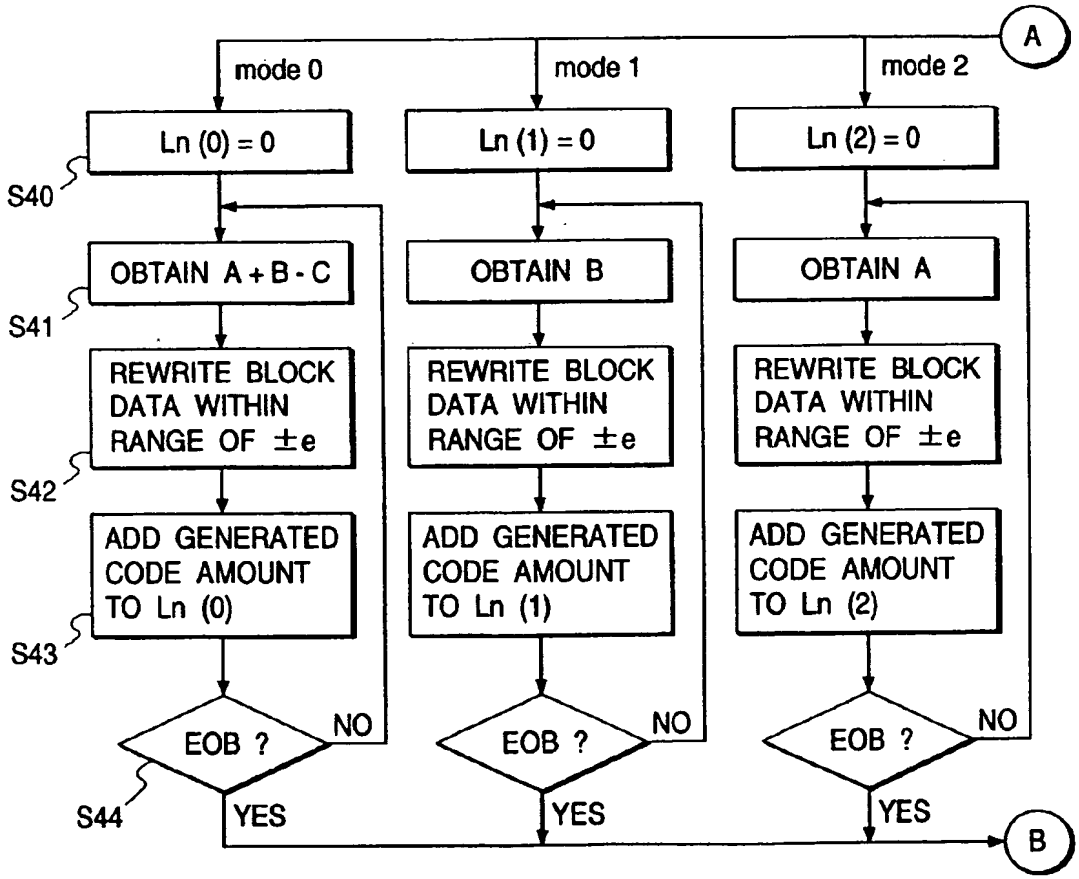


FIG. 8B

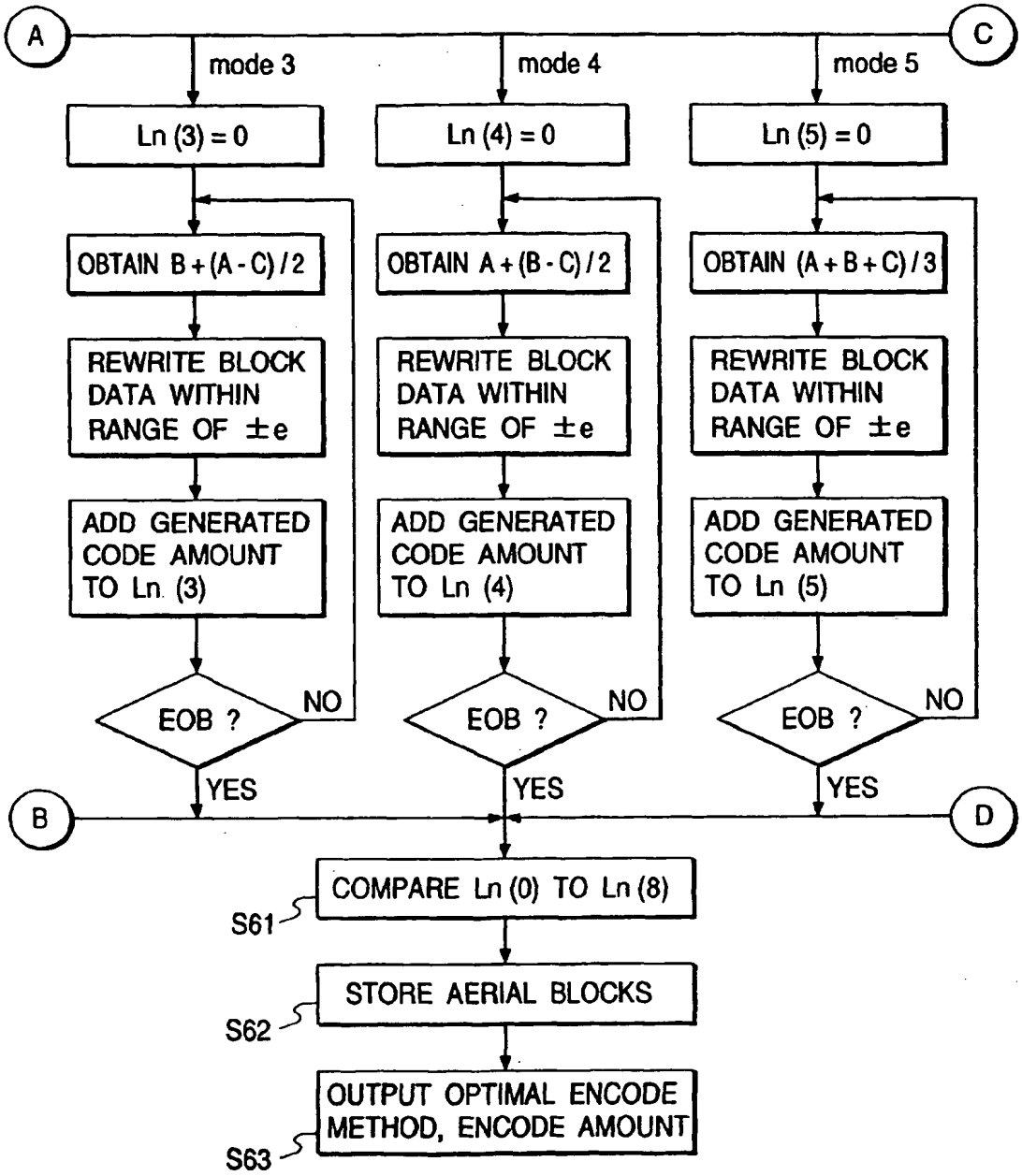

A
C
mode 3
mode 4
mode 5
Ln (3) = 0
Ln (4) = 0
Ln (5) = 0
OBTAIN B + (A - C) / 2
OBTAIN A + (B - C) / 2
OBTAIN (A + B + C) / 3
REWRITE BLOCK DATA WITHIN RANGE OF ±e
REWRITE BLOCK DATA WITHIN RANGE OF ±e
REWRITE BLOCK DATA WITHIN RANGE OF ±e
ADD GENERATED CODE AMOUNT TO Ln (3)
ADD GENERATED CODE AMOUNT TO Ln (4)
ADD GENERATED CODE AMOUNT TO Ln (5)
EOB ?
NO
YES
EOB ?
NO
YES
EOB ?
NO
YES
B
D
COMPARE Ln (0) TO Ln (8)
S61
STORE AERIAL BLOCKS
S62
OUTPUT OPTIMAL ENCODE METHOD, ENCODE AMOUNT
S63

FIG. 8C

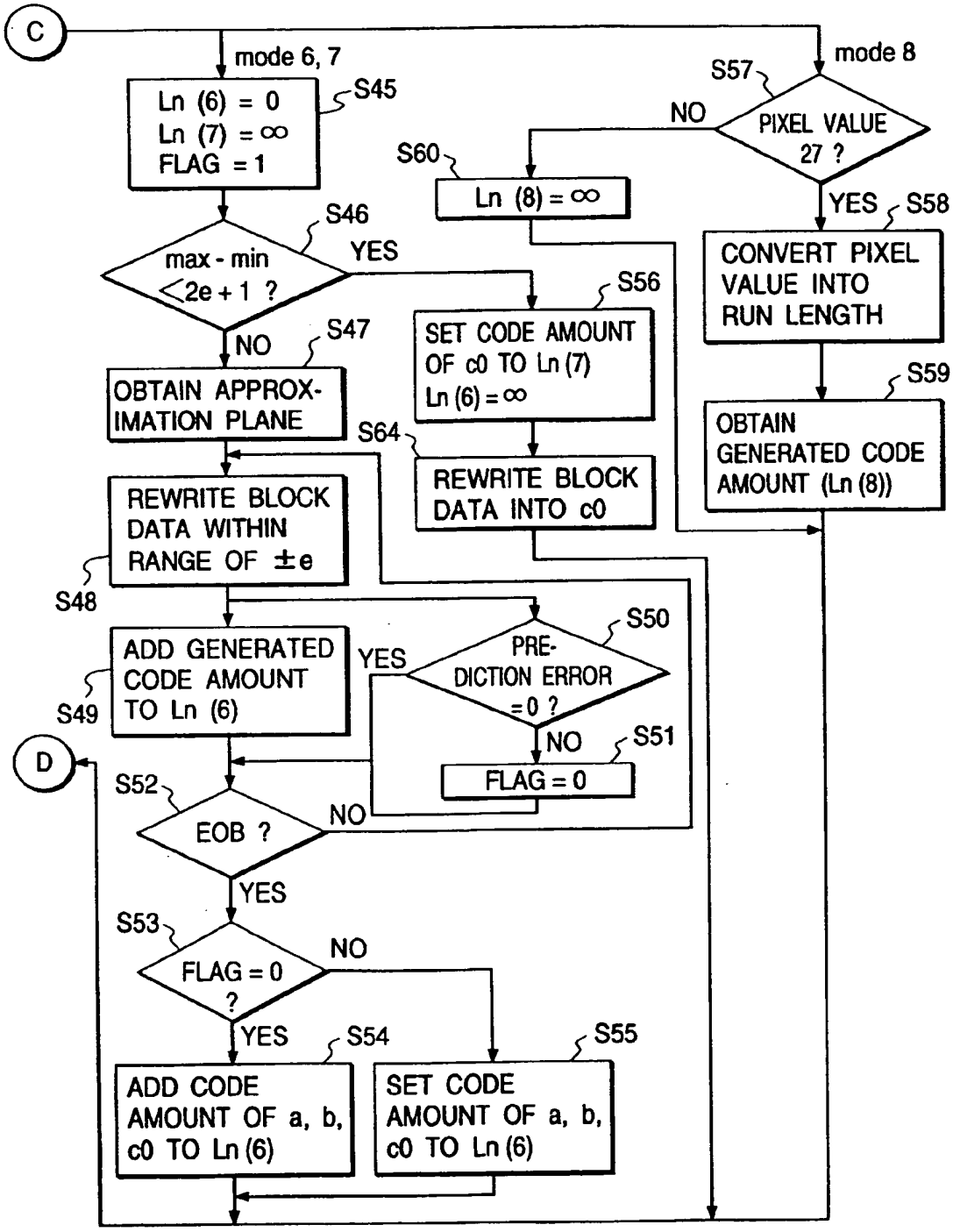

C
mode 6, 7
Ln (6) = 0
Ln (7) = ∞
FLAG = 1
S45
S46
max - min
< 2e + 1 ?
YES
NO
S47
OBTAIN APPROX-
IMATION PLANE
REWRITE BLOCK
DATA WITHIN
RANGE OF ±e
S48
ADD GENERATED
CODE AMOUNT
TO Ln (6)
S49
D
S52
EOB ?
NO
YES
S53
FLAG = 0
?
NO
YES
S54
ADD CODE
AMOUNT OF a, b,
c0 TO Ln (6)
S55
SET CODE
AMOUNT OF a, b,
c0 TO Ln (6)
S50
PRE-
DICTION ERROR
= 0 ?
YES
NO
S51
FLAG = 0
S56
SET CODE AMOUNT
OF c0 TO Ln (7)
Ln (6) = ∞
S64
REWRITE BLOCK
DATA INTO c0
S60
Ln (8) = ∞
S57
PIXEL VALUE
27 ?
NO
YES
S58
CONVERT PIXEL
VALUE INTO
RUN LENGTH
S59
OBTAIN
GENERATED CODE
AMOUNT (Ln (8))
mode 8

FIG. 9

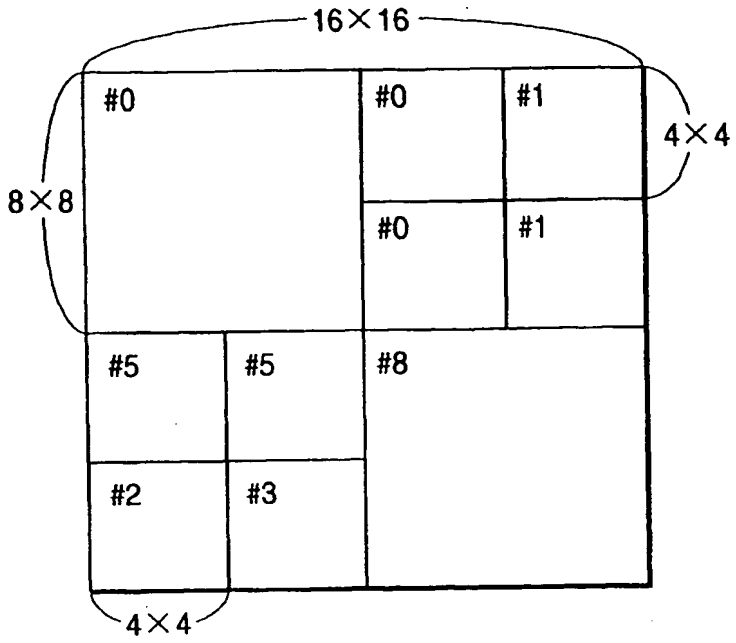

16×16
8×8
4×4
4×4
#0
#0
#1
#0
#1
#5
#5
#8
#2
#3

FIG. 10

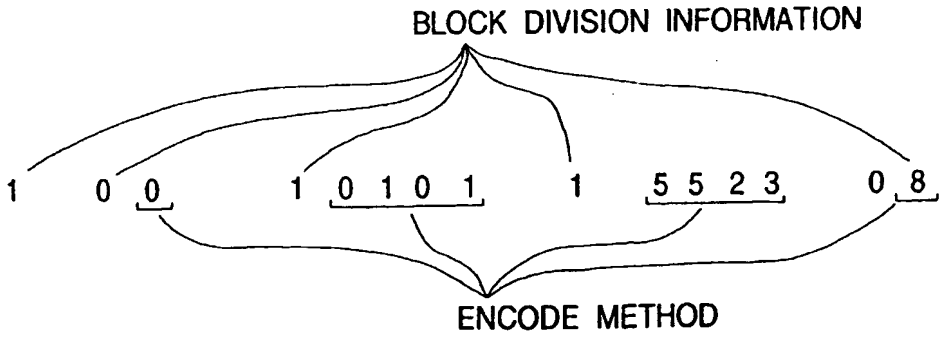

BLOCK DIVISION INFORMATION
1 0 0 1 0 1 0 1 1 5 5 2 3 0 8
ENCODE METHOD

FIG. 11

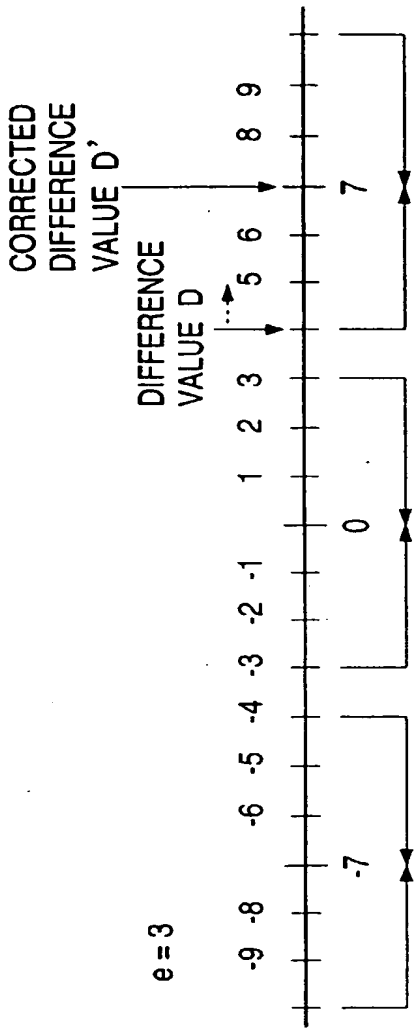

CORRECTED DIFFERENCE VALUE D'
DIFFERENCE VALUE D
e = 3
-9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 5 6 7 8 9

FIG. 12

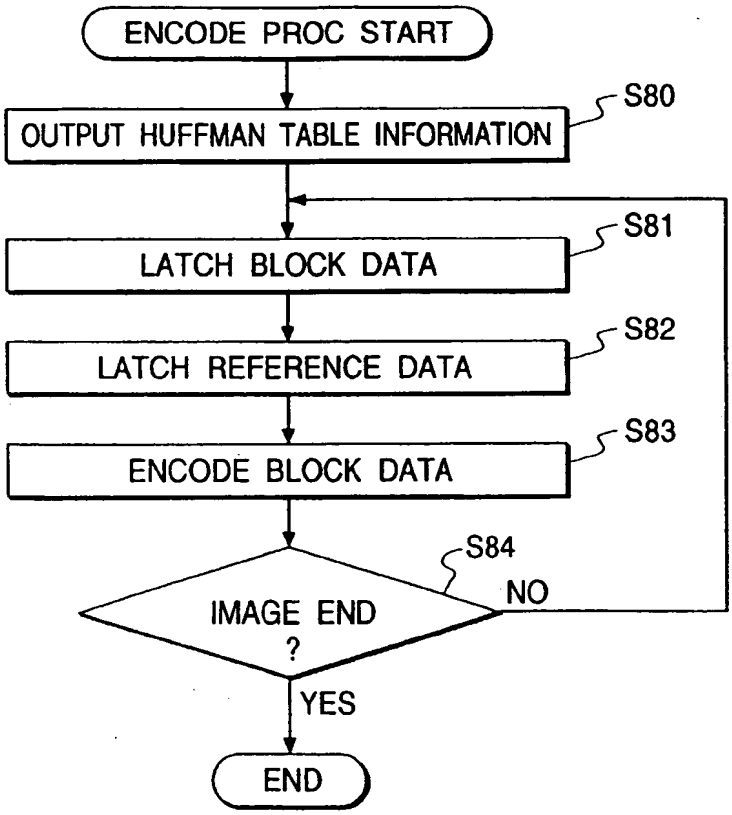

ENCODE PROC START
OUTPUT HUFFMAN TABLE INFORMATION
S80
LATCH BLOCK DATA
S81
LATCH REFERENCE DATA
S82
ENCODE BLOCK DATA
S83
IMAGE END ?
S84
NO
YES
END

FIG. 13

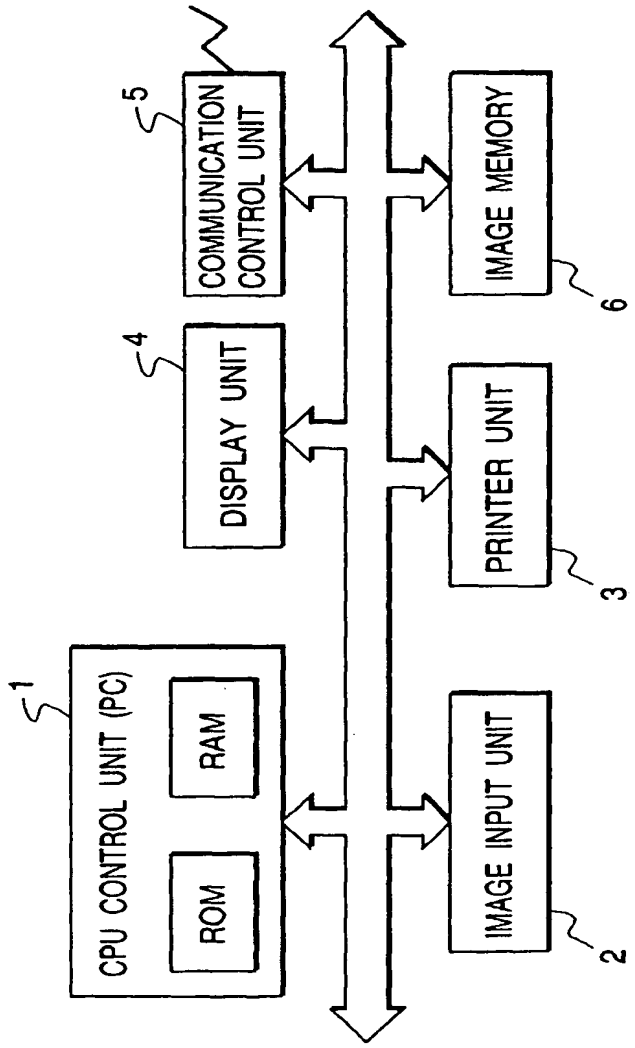

CPU CONTROL UNIT (PC)
ROM
RAM
1
DISPLAY UNIT
4
COMMUNICATION CONTROL UNIT
5
IMAGE INPUT UNIT
2
PRINTER UNIT
3
IMAGE MEMORY
6

FIG. 14

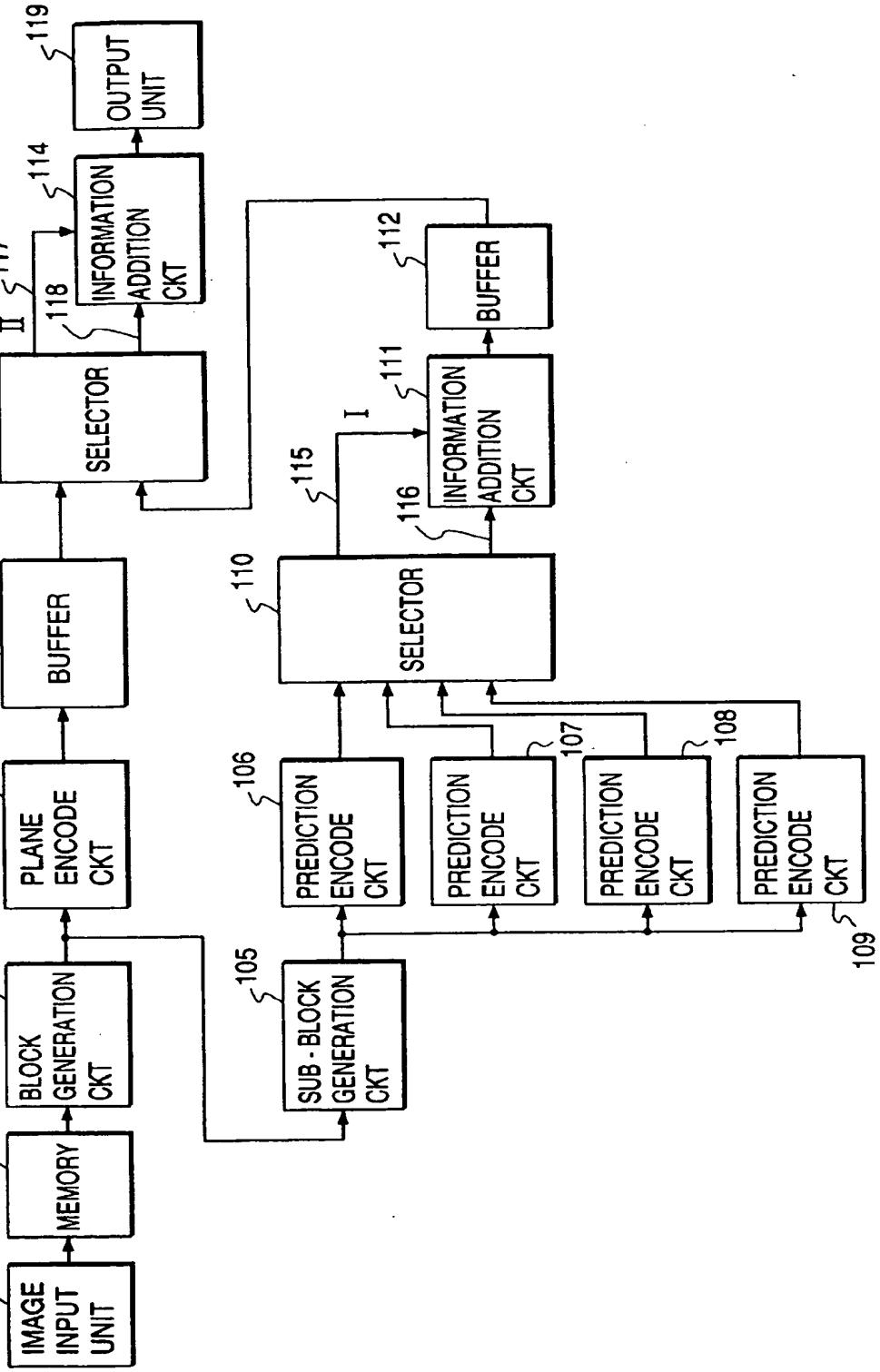


IMAGE INPUT UNIT
100
MEMORY
101
BLOCK GENERATION CKT
102
PLANE ENCODE CKT
103
BUFFER
104
SUB - BLOCK GENERATION CKT
105
PREDICTION ENCODE CKT
106
PREDICTION ENCODE CKT
107
PREDICTION ENCODE CKT
108
PREDICTION ENCODE CKT
109
SELECTOR
110
INFORMATION ADDITION CKT
111
BUFFER
112
SELECTOR
113
INFORMATION ADDITION CKT
114
OUTPUT UNIT
119
I
115
116
II
117
118

FIG. 15

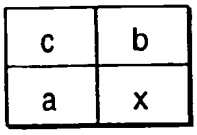


FIG. 16

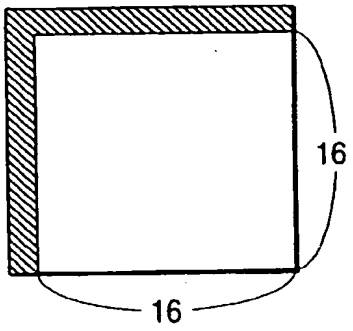
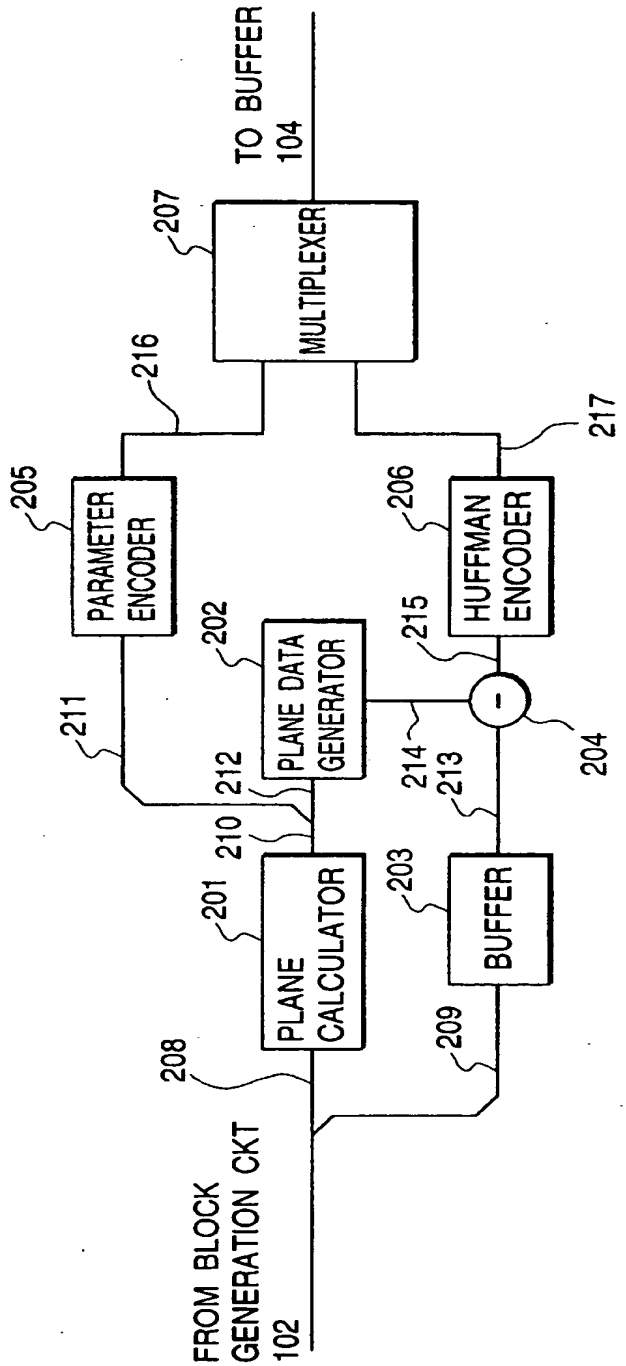


FIG. 17


TO BUFFER
104
MULTIPLEXER
207
216
217
PARAMETER ENCODER
205
HUFFMAN ENCODER
206
PLANE DATA GENERATOR
202
215
211
212
214
213
204
210
PLANE CALCULATOR
201
BUFFER
203
208
209
FROM BLOCK GENERATION CKT
102